J. Appl. Numer. Optim. 8 (2026), No. 1, pp. 115-126
Available online at http://jano.biemdas.com
https://doi.org/10.23952/jano.8.2026.1.07

A PRIMAL ALTERNATING DIRECTION METHOD OF MULTIPLIERS FOR THE COST-SENSITIVE CONSTRAINED LASSO PROBLEM

JIAJIA WANG[1], CHENGJING WANG[1,*], PEIPEI TANG[2], AIMIN XU[3], WENHAN JIA[1]

[1]*School of Mathematics, Southwest Jiaotong University, Chengdu 611731, China*
[2]*School of Computer and Computing Science, Hangzhou City University, Hangzhou 310015, China*
[3]*Institute of Mathematics, Zhejiang Wanli University, Ningbo 315100, China*

Abstract. In this paper, we investigate the cost-sensitive constrained Lasso model, a novel extension of the traditional Lasso framework that incorporates a quadratic performance constraint. This innovative approach enables simultaneous optimization of overall prediction error while rigorously maintaining prediction accuracy for designated subgroups. To address the inherent complexity arising from the nonsmooth l_1-norm regularization term in the objective function coupled with the additional quadratic performance constraint, we introduce a primal alternating direction method of multipliers (pADMM). Furthermore, we establish that pADMM achieves global convergence under mild conditions. Comprehensive numerical experiments demonstrate the effectiveness and robustness of our proposed algorithm.
Keywords. Alternating direction method of multipliers; Cost-sensitive constrained Lasso problem; Nonsmooth l_1-norm regularization.
2020 Mathematics Subject Classification. 65K05, 90C06, 90C20.

1. INTRODUCTION

Within the fields of statistics and machine learning, the Lasso (Least Absolute Shrinkage and Selection Operator) model is a fundamental regularization technique extensively employed in both linear regression and feature selection. It was first introduced in the work of Santosa and Syms [1] and was later independently rediscovered and popularized by Tibshirani [2]. Tibshirani's contributions established Lasso as a powerful tool for handling high-dimensional data, particularly in domains such as bioinformatics, economics, and social sciences. When dealing with a large number of potential explanatory variables, Lasso effectively identifies and selects the most relevant variables. The specific form of Lasso is as follows:

$$\min_{x\in\mathbb{R}^p}\left\{\frac{1}{2}\|Ax-b\|^2+\lambda\|x\|_1\right\},$$

where $A\in\mathbb{R}^{n\times p}$ represents the feature matrix, $b\in\mathbb{R}^n$ is the observation vector, and $x\in\mathbb{R}^p$ denotes the parameter vector to be estimated. The first term in the objective function corresponds to the sum of squared prediction error, which measures the goodness of fit of the model to the

*Corresponding author.
E-mail address: renascencewang@hotmail.com (C. Wang).
Received 13 June 2025; Accepted 21 August 2025; Published online 1 April 2026.

data. The second term is the l_1-norm regularization term, which penalizes the coefficients to facilitate variable selection. The regularization parameter $\lambda > 0$ controls the trade-off between these two components: a larger value of λ encourages a simpler model, potentially shrinking some coefficients to zero and thus enabling feature selection.

The effectiveness of Lasso stems from its strategic imposition of an l_1-norm penalty on the regression coefficients, which simultaneously accomplishes two critical functions: it shrinks coefficients of less significant predictors to exactly zero, thereby inducing solution sparsity, while effectively managing model complexity. This simultaneous induction of sparsity and complexity control constitutes the primary advantage of Lasso, enabling it to effectively address overfitting phenomena and significantly enhance the generalization ability of the model through the elimination of irrelevant features.

Building upon the foundational Lasso framework, Tibshirani and Taylor [3] introduced the generalized Lasso model, a significant extension that incorporates a flexible penalty matrix B to enhance model adaptability. The generalized Lasso model is formally expressed as

$$\min_{x\in\mathbb{R}^p} \left\{ \frac{1}{2}\|Ax-b\|^2 + \lambda \|Bx\|_1 \right\},$$

where the penalty matrix $B \in \mathbb{R}^{s\times p}$ serves as a design parameter that can be tailored to specific problem requirements. This formulation enables the induced sparsity pattern in Bx to precisely reflect the desired structural constraints on the parameter vector x, thereby providing greater flexibility in regularization compared to the standard Lasso framework.

Many real-world problems, such as medical diagnosis or bank credit risk assessment, may have vastly different consequences for misclassification across different categories (e.g., patients, high-risk clients). In [4, 5, 6, 7, 8], the classification accuracy for target individuals was improved by modifying classifier structures or introducing additional parameters. Blanquero et al. [9] introduced the cost-sensitive constrained Lasso (CSCLasso) model as an advanced extension of the traditional Lasso framework. Specifically designed to address subgroup-specific predictive performance, CSCLasso integrates a quadratic performance constraint to explicitly control the prediction error for designated subgroups. This innovative formulation ensures that the predictive accuracy for these subgroups strictly satisfies predefined quality thresholds while simultaneously maintaining the overall regularization benefits of the Lasso approach. The equivalent form of the CSCLasso optimization problem in [9] is formally formulated as:

$$\begin{aligned} \min_{x\in\mathbb{R}^p} \ & \frac{1}{n_0}\|A_0x-b_0\|^2 + \lambda\|Bx\|_1 \\ \text{s.t.} \ & \begin{cases} \|A_1x-b_1\| \le \sqrt{n_1 f_1}, \\ \qquad\vdots \\ \|A_mx-b_m\| \le \sqrt{n_m f_m}, \end{cases} \end{aligned} \tag{1.1}$$

where $A_i \in \mathbb{R}^{n_i\times p}$ and $b_i \in \mathbb{R}^{n_i}$ denote the feature matrix and response vector for the i-th subgroup, f_i denotes the predefined upper bound on the prediction error for the corresponding subgroup, n_i denotes the sample size, and p denotes the number of observed features. Here $n_i > p$ and $\operatorname{rank}(A_i) = p$. This innovative formulation enables CSCLasso to simultaneously achieve three fundamental objectives: (a) maintaining the overall goodness-of-fit of the model

through the primary optimization term, (b) preserving solution sparsity via the l_1-norm regularization, and (c) ensuring controlled predictive accuracy for each specific subgroup through the quadratic constraints.

Consequently, CSCLasso provides a comprehensive framework for the cost-sensitive regression that effectively balances the global model performance with subgroup-specific prediction quality. Similar constrained Lasso problems were also studied by Gaines and Zhou [10], Wang [11], Wang and Tang [12], and so on.

Blanquero et al. [9] demonstrated that, when regularization parameter λ is specified, the CSCLasso problem (1.1) can be reformulated as a standard quadratic programming problem, and its numerical solution can be obtained by using the quadratic programming solver in the Gurobi optimization toolkit. Furthermore, they showed that even for the simplified case with a single constraint, no closed-form solution can be derived through the Lagrangian multiplier method.

To address this limitation, we employ the primal alternating direction method of multipliers (pADMM) to solve problem (1.1), which was originally proposed by Glowinski and Marroco [13] and Gabay and Mercier [14]. We prove the global convergence of the algorithm. Numerical experiments demonstrate the effectiveness of our approach by comparing it with the dual symmetric Gauss-Seidel ADMM (d-sGS-ADMM) and Gurobi solver.

In Section 2, we present some preliminaries. In Section 3, we describe the specific algorithm details. In Section 4, we provide the convergence and convergence rate analysis of the algorithm. In Section 5, we present the numerical results. In Section 6, the last section, we conclude our paper.

Additional notations. Let $\mathbb{R}^n$ be the n-dimensional Euclidean space, and let I be the identity matrix. Given a function $f:\mathbb{R}^n\to(-\infty,+\infty]$, the conjugate function f^* is defined as $f^*(y):=\sup_x\{\langle y,x\rangle-f(x)\}$. For a given closed convex set M and a vector x, we denote the distance from x to M by

$$\operatorname{dist}(x,M):=\inf_{y\in M}\|x-y\|$$

and the Euclidean projection of x onto M by $\Pi_M(x):=\arg\min_{y\in M}\|x-y\|$. For $r>0$, we define $B_q^r:=\{x\in\mathbb{R}^n\mid\|x\|_q\le r\}$, where $q=1,2,\text{or }\infty$. For a set C, we define the indicator function $\delta_C(x)$ as follows: when $x\in C$, $\delta_C(x):=0$, and otherwise $\delta_C(x):=\infty$.

2. Preliminaries

In this section, we provide some necessary preliminaries. The relevant details can be found in [15, 16].

Given a function $f:\mathbb{R}^n\to[-\infty,+\infty]$, let

$$\operatorname{dom} f:=\{x\in\mathbb{R}^n\mid f(x)<+\infty\}.$$

If dom f is non-empty and $f>-\infty$, then the function f is said to be proper. In a finite-dimensional Hilbert space $\mathscr{X}$, for any positive semidefinite linear operator Γ, the distance is defined as

$$\operatorname{dist}_\Gamma(x,\mathscr{M}):=\inf_{x'\in\mathscr{M}}\|x-x'\|_\Gamma,\quad\forall x\in\mathscr{X},\ \mathscr{M}\subset\mathscr{X},$$

where $\|x\|_\Gamma := \sqrt{\langle x, \Gamma x\rangle}$, and $\langle\cdot,\cdot\rangle$ denotes the inner product. Given a function $f : \mathbb{R}^n \to [-\infty, +\infty]$, if $\liminf_{x\to x_0} f(x) = f(x_0)$, then f is said to be lower semicontinuous at x_0. Generally, if f is lower semicontinuous at every point, then f is called a lower semicontinuous function.

Given a proper lower semicontinuous convex function $f : \mathbb{R}^n \to (-\infty, +\infty]$ and a parameter $\sigma > 0$, the proximal operator $\mathrm{Prox}_{\sigma f}(\cdot)$ is defined as

$$\mathrm{Prox}_{\sigma f}(x) := \arg\min_y \left\{ f(y) + \frac{1}{2\sigma}\|y - x\|_2^2 \right\}, \quad \forall x \in \mathbb{R}^n.$$

Given the conjugate function f^*, the proximal operators $\mathrm{Prox}_{\sigma f}(\cdot)$ and $\mathrm{Prox}_{\sigma f^*}(\cdot)$ satisfies the Moreau decomposition theorem, that is,

$$\mathrm{Prox}_{\sigma f}(x) + \sigma \mathrm{Prox}_{f^*/\sigma}(\frac{x}{\sigma}) = x, \quad \forall x \in \mathbb{R}^n.$$

3. Primal ADMM

In this section, we mainly provide a detailed description of the algorithm. By introducing the auxiliary variables $z_0, z_1, \cdots, z_m, \tilde{z}$, problem (1.1) can be equivalently written as

$$\begin{aligned} \min_{\substack{x,\tilde{z}\in\mathbb{R}^p, z_0\in\mathbb{R}^{n_0},\\ z_1,\cdots,z_m\in\mathbb{R}^{n_i}}} & \frac{1}{n_0}\|z_0\|^2 + \lambda\|\tilde{z}\|_1 + \sum_{i=1}^m \delta_{B_2^{r_i}}(z_i) \\ \text{s.t.} & \begin{cases} A_0 x - z_0 - b_0 = 0, \\ A_i x - z_i - b_i = 0,\ i = 1,\cdots,m \\ Bx - \tilde{z} = 0, \end{cases} \end{aligned} \tag{3.1}$$

where $r_i = \sqrt{n_i f_i}$.

The dual of problem (3.1) is

$$\begin{aligned} \min_{\substack{w\in\mathbb{R}^{n_0}, v, \hat{v}\in\mathbb{R}^p,\\ u_1,\cdots,u_m,\hat{u}_1,\cdots,\hat{u}_m\in\mathbb{R}^{n_i}}} & \frac{n_0}{4}\|w\|^2 + \sum_{i=1}^m r_i\|\hat{u}_i\| + \langle w, b_0\rangle + \sum_{i=1}^m \langle u_i, b_i\rangle + \delta_{B_\infty^\lambda}(\hat{v}) \\ \text{s.t.} & \begin{cases} A_0^T w + \sum_{i=1}^m A_i^T u_i + B^T v = 0, \\ \hat{u}_i - u_i = 0, i = 1,\cdots,m \\ \hat{v} - v = 0. \end{cases} \end{aligned} \tag{3.2}$$

The Karush-Kuhn-Tucker (KKT) condition for problem (3.1) is

$$\begin{cases} A_0 x - z_0 - b_0 = 0, \\ A_i x - z_i - b_i = 0,\ i = 1,\cdots,m \\ Bx - \tilde{z} = 0, \\ A_0^T w + \sum_{i=1}^m A_i^T u_i + B^T v = 0, \\ \frac{2}{n_0} z_0 - w = 0, \\ u_i \in \partial \delta_{B_2^{r_i}}(z_i),\ i = 1,\cdots,m \\ v \in \lambda \partial \|\tilde{z}\|_1. \end{cases}$$

Given a parameter $\sigma > 0$, the augmented Lagrangian function for problem (3.1) is

$$\begin{aligned}
&L_\sigma(x, z_0, z_1, \cdots, z_m, \tilde{z}; w, u_1, \cdots, u_m, v) \\
=&\frac{1}{n_0}\|z_0\|^2 + \lambda\|\tilde{z}\|_1 + \sum_{i=1}^{m} \delta_{B_2^{r_i}}(z_i) \\
+&\langle w, A_0 x - z_0 - b_0\rangle + \sum_{i=1}^{m}\langle u_i, A_i x - z_i - b_i\rangle + \langle v, Bx - \tilde{z}\rangle \\
+&\frac{\sigma}{2}\|A_0 x - z_0 - b_0\|^2 + \sum_{i=1}^{m}\frac{\sigma}{2}\|A_i x - z_i - b_i\|^2 + \frac{\sigma}{2}\|Bx - \tilde{z}\|^2.
\end{aligned}$$

At the k-th iteration of pADMM, we need to minimize the augmented Lagrangian function $L_\sigma(x, z_0, z_1, \cdots, z_m, \tilde{z}; w, u_1, \cdots, u_m, v)$ with respect to the variables $x, z_0, z_1, \cdots, z_m$ and $\tilde{z}$. The key details are as follows:

• For the variable x^{k+1}:

$$\begin{aligned}
x^{k+1} = &(A_0^T A_0 + \sum_{i=1}^{m} A_i^T A_i + B^T B)^{-1} \\
&(\frac{1}{\sigma}(-A_0^T w^k - \sum_{i=1}^{m} A_i^T u_i^k - B^T v^k) + A_0^T(z_0^k + b_0) + \sum_{i=1}^{m} A_i^T(z_i^k + b_i) + B^T \tilde{z}^k).
\end{aligned}$$

• For the variable z_0^{k+1}:

$$z_0^{k+1} \quad = (w^k + \sigma(A_0 x^{k+1} - b_0))/(\tfrac{2}{n_0} + \sigma).$$

• For the variables $z_i^{k+1}, i = 1, \cdots, m$:

$$\begin{aligned}
z_i^{k+1} &= \arg\min_{z_i}\left\{\delta_{B_2^{r_i}}(z_i) - \langle u_i^k, z_i\rangle + \frac{\sigma}{2}\|A_i x^{k+1} - z_i - b_i\|^2\right\} \\
&= \arg\min_{z_i}\left\{\delta_{B_2^{r_i}}(z_i) + \frac{\sigma}{2}\|z_i - \frac{u_i^k}{\sigma} - A_i x^{k+1} + b_i\|^2\right\} \\
&= \Pi_{B_2^{r_i}}(\frac{u_i^k}{\sigma} + A_i x^{k+1} - b_i),\ i = 1, \cdots, m.
\end{aligned}$$

• For the variable $\tilde{z}^{k+1}$:

$$\begin{aligned}
\tilde{z}^{k+1} &= \arg\min_{\tilde{z}}\left\{\lambda\|\tilde{z}\|_1 - \langle v^k, \tilde{z}\rangle + \frac{\sigma}{2}\|Bx^{k+1} - \tilde{z}\|^2\right\} \\
&= \arg\min_{\tilde{z}}\left\{\lambda\|\tilde{z}\|_1 + \frac{\sigma}{2}\|\tilde{z} - \frac{\hat{v}^k}{\sigma} - Bx^{k+1}\|^2\right\} \\
&= \mathrm{Prox}_{\lambda/\sigma\|\cdot\|_1}(\frac{\hat{v}^k}{\sigma} + Bx^{k+1}).
\end{aligned}$$

In the following, we provide the algorithm framework of pADMM for problem (3.1).

Algorithm 1 (pADMM for problem (3.1)):
Set $\sigma > 0,\ \tau = 1.618$. Let $k = 0$, and input the initial point $(x^0, z_0^0, z_1^0, \cdots, z_m^0, \tilde{z}^0, w^0, u_1^0, \cdots, u_m^0, v^0)$.
Step 1: Update x^{k+1}:

$$x^{k+1} = (A_0^T A_0 + \sum_{i=1}^{m} A_i^T A_i + B^T B)^{-1} \\ ((-A_0^T w^k - \sum_{i=1}^{m} A_i^T u_i^k - B^T v^k)/\sigma + A_0^T(z_0^k + b_0) + \sum_{i=1}^{m} A_i^T(z_i^k + b_i) + B^T \tilde{z}^k).$$

Step 2: Update $z_0^{k+1}, z_i^{k+1}, \tilde{z}^{k+1}$:

$$z_0^{k+1} = (w^k + \sigma(A_0 x^{k+1} - b_0))/(2/n_0 + \sigma),$$
$$z_i^{k+1} = \Pi_{B_2^{r_i}}(u_i^k/\sigma + A_i x^{k+1} - b_i),\ i = 1, \cdots, m$$
$$\tilde{z}^{k+1} = \mathrm{Prox}_{\lambda/\sigma\|\cdot\|_1}(v^k/\sigma + B x^{k+1}).$$

Step 3: Update $w^{k+1}, u_i^{k+1}, v^{k+1}$:

$$w^{k+1} = w^k + \tau\sigma(A_0 x^{k+1} - z_0^{k+1} - b_0),$$
$$u_i^{k+1} = u_i^k + \tau\sigma(A_i x^{k+1} - z_i^{k+1} - b_i),\ i = 1, \cdots, m$$
$$v^{k+1} = v^k + \tau\sigma(B x^{k+1} - \tilde{z}^{k+1}).$$

Step 4: The algorithm terminates when the stopping criterion is satisfied; otherwise, set $k = k + 1$, and returns to Step 1.

4. Convergence Analysis

In this section, we present the convergence analysis of pADMM. Let $f(x) = 0$, and

$$g(z_0, z_1, \cdots, z_m, \tilde{z}) = \frac{1}{n_0}\|z_0\|^2 + \lambda\|\tilde{z}\|_1 + \sum_{i=1}^{m} \delta_{B_2^{r_i}}(z_i),$$

where f and g are closed, proper convex functions. For technical reasons, we consider the following constraint qualification (**CQ**):

CQ: There exists $(x^0, z_0^0, z_1^0, \cdots, z_m^0, \tilde{z}^0) \in \mathrm{ri}(\mathrm{dom} f \times \mathrm{dom} g) \cap P$, where P is the constraint set in (3.1).

Bsaed on [17, Theorem B.1], we have the following convergence result.

Theorem 4.1. *Assume that the solution set of (3.1) is nonempty and that the **CQ** holds. Let $\tau \in (0, \frac{1+\sqrt{5}}{2})$. Let $\{(x^k, z_0^k, z_1^k, \cdots, z_m^k, \tilde{z}^k, w^k, u_1^k, \cdots, u_m^k, v^k)\}$ be the sequence generated by pADMM. Then $\{x^k, z_0^k, z_1^k, \cdots, z_m^k, \tilde{z}^k\}$ converges to the optimal solution of the original problem (3.1), and $\{(w^k, u_1^k, \cdots, u_m^k, v^k)\}$ converges to the optimal solution of the dual problem (3.2).*

Proof. Under the **CQ**, it follows from [15, Corollaries 28.2.2 and 28.3.1] that $(\bar{x}, \bar{z}_0, \bar{z}_1, \cdots, \bar{z}_m, \bar{\tilde{z}})$ is an optimal solution to problem (3.1) if and only if there exists a Lagrange multiplier $(\bar{w}, \bar{u}_1, \cdots, \bar{u}_m, \bar{v})$ such that

$$\begin{cases} A_0\bar{x}-\bar{z}_0-b_0=0,\ A_i\bar{x}-\bar{z}_i-b_i=0,\ i=1,\cdots,m,\\ B\bar{x}-\bar{\tilde{z}}=0,\ A_0^T\bar{w}+\sum_{i=1}^m A_i^T\bar{u}_i+B^T\bar{v}=0,\\ \frac{2}{n_0}\bar{z}_0-\bar{w}=0,\\ \bar{u}_i\in\partial\delta_{B_2^{r_i}}(\bar{z}_i),\ i=1,\cdots,m,\ \bar{v}\in\lambda\partial\|\bar{\tilde{z}}\|_1. \end{cases}$$

where $A_0\in\mathbb{R}^{n_0\times p}$ and $\mathrm{rank}(A_0)=p$. Based on the theoretical framework established in [17, Theorem B.1], we only need to make sure that the assumptions hold. Since

$$\sigma A_0^TA_0+\sigma\sum_{i=1}^m A_i^TA_i+\sigma B^TB\succ 0,\quad \begin{pmatrix} \frac{2}{n_0}I+\sigma I & & & \\ & \ddots & & \\ & & \sigma I & \\ & & & \sigma I \end{pmatrix}\succ 0,$$

the desired result follows. □

5. Numerical Experiments

In this section, we present the numerical experiment results of the proposed algorithm. We focus on comparing the numerical performances of pADMM, d-sGS-ADMM and Gurobi solver. All experiments were conducted on a 64-bit Windows 10 laptop with the following specifications: an AMD Ryzen 7 5700U processor with Radeon Graphics @1.80GHz, 16GB of RAM, and MATLAB 2021a as the runtime environment.

The iteration stops if the algorithms satisfy $\eta_{KKT}:=\max\{R_p,R_d,R_c\}\le 10^{-6}$, where the residuals R_p, R_d, and R_c are defined as:

$$R_p=\frac{\|A_0x-z_0-b_0\|+\sum_{i=1}^m\|A_ix-z_i-b_i\|+\|Bx-\tilde{z}\|}{1+\|z_0\|+\sum_{i=1}^m\|z_i\|+\|\tilde{z}\|},$$
$$R_d=\frac{\|A_0^Tw+\sum_{i=1}^m A_i^Tu_i+B^Tv\|+\|\frac{2}{n_0}z_0-w\|}{1+\|x\|+\|z_0\|},$$
$$R_c=\frac{\sum_{i=1}^m\|z_i-\Pi_{B_2^{r_i}}(z_i+u_i)\|+\|\tilde{z}-\mathrm{Prox}_{\lambda\|\cdot\|_1}(\tilde{z}+v)\|}{1+\sum_{i=1}^m\|z_i\|+\|\tilde{z}\|}.$$

The maximal iteration number for pADMM and d-sGS-ADMM is 30,000.

The random data is generated according to the following code:

```
x_real = randn(p+1,1);
A0 = sortrows([ones(n0,1) zscore(10*randn(n0,p))],2);
A1 = A0(1:n1,:);
A2 = A0(1+n1:n1+n2,:);
A3 = A0(1+n1+n2:n1+n2+n3,:);
b0 = A0*x_real+0.5*randn(n0,1);
b1 = b0(1:n1);
```

$$\texttt{b2} = \texttt{b0}(1+\texttt{n1}:\texttt{n1}+\texttt{n2});$$

$$\texttt{b3} = \texttt{b0}(1+\texttt{n1}+\texttt{n2}:\texttt{n1}+\texttt{n2}+\texttt{n3});$$

$$\texttt{B} = [\texttt{zeros(p,1)}\ \texttt{eye(p,p)}];$$

The parameter $f_i = (1+\tau)\mathrm{MSE}_i(x),\ i = 1,\cdots,m$, where $\mathrm{MSE}_i(x) = \frac{1}{n_i}\|A_i x - b_i\|^2,\ i = 1,\cdots,m$ and $\tau \geq 0$. The parameter λ is selected through a five-fold cross-validation.

5.1. **d-sGS-ADMM.** For the convenience of the readers, we present the framework of d-sGS-ADMM as below:

Algorithm 2 (d-sGS-ADMM for problem (3.2)):
Set $\sigma > 0,\ \tau = 1.618$. Let $k = 0$, and input the initial point $(w^0, u_1^0, \cdots, u_m^0, v^0, \hat{u}_1^0, \cdots, \hat{u}_m^0, \hat{v}^0, x^0, z_1^0, \cdots, z_m^0, \tilde{z}^0)$.
Step 1: Update $w^{k+1}, \hat{u}_i^{k+1}, \hat{v}^{k+1}$:

$$w^{k+1} = \left(\frac{n_0}{2}I + \sigma A_0 A_0^T\right)^{-1}(A_0 x^k - b_0 - A_0(\sum_{i=1}^m A_i^T u_i^k + B^T v^k)),$$

$$\hat{u}_i^{k+1} = \mathrm{Prox}_{r_i/\sigma\|\cdot\|}\left(z_i^k/\sigma + u_i^k\right),\ i = 1,\cdots,m$$

$$\hat{v}^{k+1} = \Pi_{B_\infty^\lambda}\left(\tilde{z}^k/\sigma + v^k\right).$$

Step 2: Update u_i^{k+1}, v^{k+1}:

$$u_i^{k+\frac{1}{2}} = (I + A_i A_i^T)^{-1}((A_i x^k - z_i^k - b_i)/\sigma$$
$$- A_i(A_0^T w^{k+1} + \sum_{j=1}^{i-1} A_j^T u_j^{k+\frac{1}{2}} + \sum_{j=i+1}^{m} A_j^T u_j^k + B^T v^k) + \hat{u}_i^{k+1}),\ i = 1,\cdots,m$$

$$v^{k+1} = (I + BB^T)^{-1}((Bx^k - \tilde{z}^k)/\sigma - B(A_0^T w^{k+1} + \sum_{i=1}^{m} A_i^T u_i^{k+\frac{1}{2}}) + \hat{v}^{k+1}),$$

$$u_i^{k+1} = (I + A_i A_i^T)^{-1}((A_i x^k - z_i^k - b_i)/\sigma$$
$$- A_i(A_0^T w^{k+1} + \sum_{j=1}^{i-1} A_j^T u_j^{k+\frac{1}{2}} + \sum_{j=i+1}^{m} A_j^T u_j^{k+1} + B^T v^{k+1}) + \hat{u}_i^{k+1}),\ i = m,\cdots,1$$

Step 3: Update $x^{k+1}, z_i^{k+1}, \tilde{z}^{k+1}$:

$$x^{k+1} = x^k - \tau\sigma(A_0^T w^{k+1} + \sum_{i=1}^{m} A_i^T u_i^{k+1} + B^T v^{k+1}),$$

$$z_i^{k+1} = z_i^k - \tau\sigma(\hat{u}_i^{k+1} - u_i^{k+1}),\ i = 1,\cdots,m$$

$$\tilde{z}^{k+1} = \tilde{z}^k - \tau\sigma(\hat{v}^{k+1} - v^{k+1}).$$

Step 4: The algorithm terminates when the stopping criterion is satisfied; otherwise, set $k = k+1$, and returns to Step 1.

5.2. **Experimental results.** We focus on problem (1.1) and compare the numerical performances of pADMM, d-sGS-ADMM and Gurobi solver on different datasets. In the results, we report the dataset name, sample size (n), feature size (p), regularization parameter (λ), number of iterations (iter), relative KKT residual (KKT residual), objective function value (objective), and running time (time, in seconds). We solve problem (1.1) with a single constraint ($m = 1$) and multiple constraints ($m = 3$), respectively, and present the numerical results on random data problems in Tables 1 and 2, and the results on real data problems [18] in Tables 3 and 4. Additionally, we plot Figures 1 and 2 to provide a more intuitive comparison of the running times of the three algorithms when solving random data problems.

The results presented in Tables 1 - 4 demonstrate that pADMM can successfully solve all the problems, Gurobi can solve most of the problems. It should be noted that Gurobi's termination criterion is not solely dependent on the KKT residual, it also halts when the improvement in the objective function value becomes negligible over successive iterations. Consequently, in some instances, Gurobi fails to meet the KKT residual accuracy requirement of $\eta_{KKT} \leq 10^{-6}$. On the other hand, d-sGS-ADMM fails to obtain satisfying solutions for several problems. As can be observed from Figures 1 and 2, the running time mostly follows the order d-sGS-ADMM $>$ Gurobi $>$ pADMM. However, as the scale of the problem increases, particularly in the case of $m = 3$, the growth rate of pADMM's running time exceeds that of Gurobi.

TABLE 1. Comparison of d-sGS-ADMM, pADMM and Gurobi for the problem (1.1) with the random data ($m = 1$)

dataname(n,p)	d-sGS-ADMM|pADMM|Gurobi				
	λ	iter	KKT residual	objective	time
rand(200,20)	4.13e-3	332|113|11	9.98e-7|9.88e-7|7.32e-8	3.1134e-1|3.1113e-1|3.1113e-1	0.04|0.02|0.10
rand(400,40)	2.06e-3	611|232|11	9.97e-7|9.88e-7|1.83e-8	3.0451e-1|3.0443e-1|3.0444e-1	0.04|0.02|0.08
rand(600,60)	1.83e-3	829|141|14	9.93e-7|9.05e-7|9.10e-9	3.1541e-1|3.1530e-1|3.1530e-1	0.06|0.01|0.09
rand(800,80)	8.11e-4	1048|149|15	9.96e-7|8.32e-7|3.24e-7	2.6605e-1|2.6602e-1|2.6603e-1	0.09|0.01|0.12
rand(1000,100)	5.72e-4	1225|154|17	9.99e-7|9.92e-7|1.75e-7	2.6777e-1|2.6775e-1|2.6775e-1	0.16|0.02|0.14
rand(1200,120)	1.26e-4	1392|159|15	9.98e-7|9.97e-7|5.35e-6	2.3315e-1|2.3315e-1|2.3315e-1	0.34|0.02|0.14
rand(1400,140)	9.11e-4	1607|164|14	9.97e-7|8.69e-7|1.72e-8	3.2106e-1|3.2099e-1|3.2099e-1	0.62|0.03|0.14
rand(1600,160)	5.72e-4	1958|282|17	9.99e-7|8.85e-7|1.16e-6	3.1396e-1|3.1393e-1|3.1393e-1	1.19|0.08|0.19
rand(1800,180)	1.02e-3	1966|170|16	9.99e-7|8.26e-7|2.70e-7	3.8996e-1|3.8986e-1|3.8986e-1	1.69|0.07|0.22
rand(2000,200)	2.01e-4	2131|173|17	9.98e-7|9.58e-7|1.35e-6	2.5584e-1|2.5583e-1|2.5583e-1	2.27|0.11|0.26

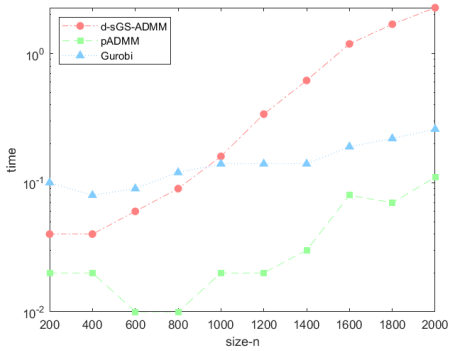

FIGURE 1. Comparison of running times of different algorithms with respect to different problems in Table 1 ($m = 1$)

TABLE 2. Comparison of d-sGS-ADMM, pADMM and Gurobi for problem (1.1) with the random data ($m = 3$)

dataname(n,p)	d-sGS-ADMM|pADMM|Gurobi				
	λ	iter	KKT residual	objective	time
rand(200,20)	1.45e-3	984|118|9	9.90e-7|8.43e-7|4.65e-7	2.6195e-1|2.6192e-1|2.6192e-1	0.08|0.03|0.09
rand(400,40)	2.01e-4	1747|136|11	9.90e-7|8.33e-7|8.50e-7	2.4527e-1|2.4527e-1|2.4527e-1	0.13|0.01|0.08
rand(600,60)	2.60e-3	2508|146|13	9.97e-7|9.57e-7|1.82e-6	3.3336e-1|3.3309e-1|3.3309e-1	0.27|0.02|0.09
rand(800,80)	1.83e-3	3274|175|12	9.84e-7|9.67e-7|3.17e-8	3.8321e-1|3.8308e-1|3.8308e-1	0.59|0.03|0.12
rand(1000,100)	7.22e-4	4279|159|17	9.96e-7|9.78e-7|3.31e-7	2.9101e-1|2.9099e-1|2.9099e-1	1.22|0.03|0.13
rand(1200,120)	8.11e-4	4899|164|15	9.97e-7|7.97e-7|7.54e-7	2.8886e-1|2.8881e-1|2.8881e-1	2.44|0.03|0.13
rand(1400,140)	1.83e-3	5899|200|16	9.97e-7|9.95e-7|2.13e-8	4.3269e-1|4.3241e-1|4.3241e-1	4.63|0.06|0.15
rand(1600,160)	9.11e-4	6814|173|18	1.00e-6|7.62e-7|1.33e-6	3.2470e-1|3.2463e-1|3.2463e-1	7.30|0.08|0.19
rand(1800,180)	1.15e-3	7534|175|15	9.95e-7|8.19e-7|2.86e-6	3.8773e-1|3.8758e-1|3.8758e-1	10.75|0.12|0.21
rand(2000,200)	3.09e-4	8163|178|17	9.98e-7|9.19e-7|1.42e-6	2.6738e-1|2.6737e-1|2.6737e-1	15.76|0.17|0.23

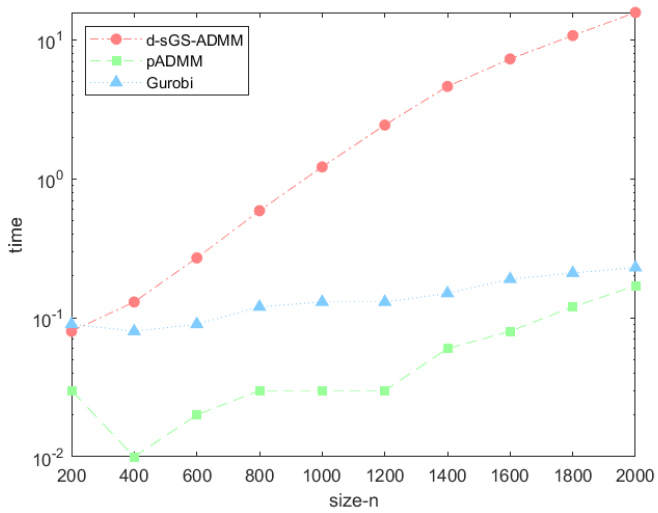

FIGURE 2. Comparison of running times of different algorithms with respect to different problems in Table 2 ($m = 3$)

TABLE 3. Comparison of d-sGS-ADMM, pADMM and Gurobi for problem (1.1) with the real data ($m = 1$)

dataname(n,p)	d-sGS-ADMM|pADMM|Gurobi				
	λ	iter	KKT residual	objective	time
CSMrating(187,11)	1.07e-1	30000|193|11	3.36e-2|9.83e-7|1.48e-7	8.6320e-1|8.1940e-1|8.1940e-1	0.55|0.02|0.09
ForestFires(243,10)	1.48e-2	602|190|8	9.87e-7|9.87e-7|5.51e-9	8.8864e-2|8.3873e-2|8.3873e-2	0.04|0.02|0.08
ConcreteStrength(1030,8)	2.60e-3	1723|158|15	9.98e-7|8.67e-7|8.53e-7	1.0732e2|1.0732e2|1.0732e2	0.07|0.01|0.08
MusicLatitude(1059,68)	1.56e0	30000|1438|15	1.18e-1|9.97e-7|1.02e-7	3.8243e2|2.9177e2|2.9176e2	2.64|0.08|0.12
Scaledsoundpressurelevel(1503,5)	2.92e-3	1778|167|18	9.98e-7|9.90e-7|9.93e-5	2.3072e1|2.3072e1|2.3072e1	0.08|0.01|0.34
winequalityred(1599,12)	2.36e-2	30000|732|17	3.02e-3|9.97e-7|6.56e-8	4.4390e-1|4.3896e-1|4.3896e-1	1.28|0.03|0.09
CommunitiesCrime(1993,100)	6.43e-4	4449|610|9	1.00e-6|9.73e-7|3.42e-9	1.7971e-2|1.7394e-2|1.7394e-2	1.64|0.08|0.11
CustomerChurn(3150,13)	5.09e-4	3250|180|9	1.00e-6|8.72e-7|2.29e-8	7.3553e-2|7.3533e-2|7.3533e-2	0.28|0.02|0.08
tempforecastmax(7590,22)	5.21e-3	30000|348|21	1.81e-4|8.00e-7|1.27e-6	2.1591e0|2.1588e0|2.1588e0	9.53|0.08|0.11
AirTemperature(8991,9)	3.20e-4	13747|217|14	1.00e-6|9.16e-7|9.36e-8	2.9945e1|2.9945e1|2.9945e1	3.13|0.04|0.08

TABLE 4. Comparison of d-sGS-ADMM, pADMM and Gurobi for problem (1.1) with the real data ($m = 3$)

dataname(n,p)	d-sGS-ADMM|pADMM|Gurobi				
	λ	iter	KKT residual	objective	time
CSMrating(187,11)	6.73e-2	30000|186|14	2.92e-2|9.61e-7|3.46e-8	8.3296e-1|7.8028e-1|7.8027e-1	1.20|0.03|0.09
ForestFires(243,10)	9.33e-3	1638|201|8	9.76e-7|9.46e-7|2.03e-8	8.2265e-2|7.9866e-2|7.9866e-2	0.11|0.04|0.09
ConcreteStrength(1030,8)	1.67e-2	3790|157|14	9.74e-7|9.84e-7|8.72e-6	1.0791e2|1.0791e2|1.0791e2	0.31|0.01|0.08
MusicLatitude(1059,68)	6.14e-1	30000|914|17	9.45e-3|9.94e-7|5.13e-8	3.1561e2|2.7394e2|2.7394e2	5.25|0.08|0.12
Scaledsoundpressurelevel(1503,5)	8.30e-3	6692|169|18	9.40e-7|9.76e-7|8.18e-5	2.3144e1|2.3144e1|2.3144e1	0.52|0.01|0.09
winequalityred(1599,12)	1.67e-2	30000|475|15	2.54e-3|9.41e-7|9.07e-8	4.3598e-1|4.3296e-1|4.3296e-1	2.53|0.04|0.09
CommunitiesCrime(1993,100)	5.72e-4	11878|326|9	9.95e-7|9.81e-7|5.72e-9	1.7942e-2|1.7319e-2|1.7319e-2	8.96|0.07|0.11
CustomerChurn(3150,13)	1.29e-3	10597|261|10	9.75e-7|9.63e-7|1.89e-8	7.4093e-2|7.3982e-2|7.3982e-2	1.68|0.04|0.08
tempforecastmax(7590,22)	3.27e-3	30000|357|22	1.39e-4|9.42e-7|3.51e-7	2.1482e0|2.1480e0|2.1480e0	20.98|0.10|0.10
AirTemperature(8991,9)	1.02e-3	24536|217|15	9.94e-7|8.16e-7|1.12e-7	2.9969e1|2.9969e1|2.9969e1	10.91|0.06|0.08

6. CONCLUSION

In this paper, we fully exploited the special structure of the CSCLasso problem and designed a pADMM. In addition, we established the global convergence theory of the algorithm under mild conditions. Finally, numerical experiments were conducted to demonstrate the effectiveness of the proposed algorithm.

Acknowledgements

We sincerely thank the Editor-in-Chief Professor Xiaolong Qin and the anonymous reviewer for their valuable comments, which greatly improved the quality of this paper. Chengjing Wang's work was supported in part by Zhejiang Provincial Natural Science Foundation of China (Grant No. LTGY23H240002). Peipei Tang's work was supported in part by Zhejiang Provincial Natural Science Foundation of China (Grant No. LMS26A010022). Aimin Xu's work was supported in part by Zhejiang Provincial Natural Science Foundation of China (Grant No. LTGY23H240002).

REFERENCES

[1] F. Santosa, W. W. Symes, Linear inversion of band-limited reflection seismograms, SIAM J. Sci. Stat. Comput. 7 (1986), 1307–1330.

[2] R. Tibshirani, Regression shrinkage and selection via the Lasso, J. Roy. Statist. Soc. Ser. B 58 (1996), 267–288.

[3] R. J. Tibshirani, J. Taylor, The solution path of the generalized Lasso, Ann. Statist. 39 (2011), 1335–1371.

[4] J. Bradford, C. Kunz, R. Kohavi, C. Brunk, C. Brodley, Pruning decision trees with misclassification costs, in: C. Nedellec, C. Rouveirol (Eds.), Machine Learning: ECML-98, pp. 131-136, Springer Berlin Heidelberg, Berlin, Heidelberg, 1998.

[5] A. Freitas, A. Costa-Pereira, P. Brazdil, Cost-sensitive decision trees applied to medical data, In: I. Song, J. Eder, T. Nguyen (Ed.), Data Warehousing and Knowledge Discovery, pp. 303-312, Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[6] E. Carrizosa, B. Martin-Barragan, D. Morales, Multi-group support vector machines with measurement costs: A biobjective approach, Discrete Appl. Math. 156 (2008), 950-966.

[7] S. Datta, S. Das, Near-Bayesian support vector machines for imbalanced data classification with equal or unequal misclassification costs, Neural Netw. 70 (2015), 39-52.

[8] W. Lee, C.-H. Jun, J.-S. Lee, Instance categorization by support vector machines to adjust weights in AdaBoost for imbalanced data classification, Inform. Sci. 381 (2017), 92-103.

[9] R. Blanquero, E. Carrizosa, P. Ramírez-Cobo, M. R. Sillero-Denamiel, A cost-sensitive constrained Lasso, Adv. Data Anal. Classif. 15 (2021) 121-158.
[10] B. R. Gaines, H. Zhou, Algorithms for fitting the constrained Lasso, J. Comput. Graph. Statist. 27 (2018), 861-871.
[11] Q. Wang, A dual alternating direction method of multipliers for the constrained Lasso problems, in: Proceedings of the 2nd International Conference on Algorithms, Computing and Systems, pp. 42-47, ACM, Beijing, 2018.
[12] C. Wang, P. Tang, A dual semismooth Newton based augmented Lagrangian method for large-scale linearly constrained sparse group square-root Lasso problems, J. Sci. Comput. 96 (2023), 45.
[13] R. Glowinski, A. Marroco, Sur l'approximation, par éléments finis d'ordre un, et la résolution, par pénalisation-dualité d'une classe de problèmes de dirichlet non linéaires, RAIRO Anal. Numér. 9 (1975), 41-76.
[14] D. Gabay, B. Mercier, A dual algorithm for the solution of nonlinear variational problems via finite element approximation, Comput. Math. Appl. 2 (1976), 17-40.
[15] R. T. Rockafellar, Convex Analysis, Princeton University Press, Princeton, 1970.
[16] R. T. Rockafellar, R. J.-B. Wets, Variational Analysis, Springer, 1998.
[17] M. Fazel, T. K. Pong, D. Sun, P. Tseng, Hankel matrix rank minimization with applications to system identification and realization, SIAM J. Matrix Anal. Appl. 34 (2013), 946-977.
[18] M. Kelly, R. Longjohn, K. Nottingham, The UCI machine learning repository, https://archive.ics.uci.edu, accessed: 19 April 2025 (2023).